IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Morgan**	§	
	§	Group Art Unit: **2192**
Serial No. **10/612,457**	§	
	§	Examiner: **Isaac Tuku Tecklu**
Filed: **July 2, 2003**	§	
	§	
For: **Method, Apparatus, and Program for Code Reusability and Maintainability in XML-Driven Projects**	§ §	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

35525
PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Notice of Appeal, filed in this case on January 9, 2008.

A fee of $510.00 is required for filing an Appeal Brief. Please charge this fee to IBM Corporation Deposit Account No. 09-0447. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation of Armonk, New York.

RELATED APPEALS AND INTERFERENCES

This appeal has no related proceedings or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

The claims in the application are: 1, 3-23

B. STATUS OF ALL THE CLAIMS IN APPLICATION

Claims canceled: 2;

Claims withdrawn from consideration but not canceled: NONE

Claims pending: 1, 3-23;

Claims allowed: NONE;

Claims rejected: 1, 3-23;

Claims objected to: NONE.

C. CLAIMS ON APPEAL

The claims on appeal are: 1, 3-23

STATUS OF AMENDMENTS

No amendments were submitted after the Final Office Action of November 27, 2007.

SUMMARY OF CLAIMED SUBJECT MATTER

A. CLAIM 1 - INDEPENDENT

The subject matter of claim 1 is directed to a method, in a data processing system, for code reusability and maintainability (Specification, p.3, ll.3-23; Figure 4, 422; Figure 5, 522). The method includes providing a utility class in a server wherein the utility class defines a utility method (Specification, p. 11, l. 10-p. 12, l. 2). The method also includes, responsive to receiving a request at the server for attributes for an entity from a client (Specification, p. 11, ll. 10-20; Figure 4, "XML Request") generating a method call for the utility method (Specification, p. 12, ll. 3-4; Figure 4 "Call Method"), wherein the method call identifies the entity and a response object name (Specification, p. 14, ll. 11-24; Figure 5). The method finally includes generating a response object (Specification, p. 14, ll. 11-24; Figure 4, "XML Response Object") and assigning the response object name to the response object (Specification, p. 14, ll. 11-24). The method further includes returning the response object to the client (Specification, p. 15, ll. 2-4).

B. CLAIM 5 – DEPENDENT

The subject matter of claim 5 is directed to the method, in a data processing system, for code reusability and maintainability of claim 1. The method of claim 5 further includes retrieving, at least one data item for the method call and the entity (Specification, p. 12, ll. 3-27), wherein the response object includes at least one data item (Specification, p. 12, ll. 17-27).

C. CLAIM 8 – DEPENDENT

The subject matter of claim 8 is directed to the method, in a data processing system, for code reusability and maintainability of claim 1. The method of claim 8 further includes the request to include a list of attributes (Specification, p. 12, ll. 17-21; p. 14, ll. 14-24).

D. CLAIM 13 – INDEPENDENT

The subject matter of claim 13 is directed to an apparatus, in a data processing system, for code reusability and maintainability (Specification, p.3, ll.3-23; Figure 4, 422; Figure 5, 522). The apparatus includes a utility class, wherein the utility class defines a utility method (Specification, p. 11, l. 10-p. 12, l. 2). The apparatus further includes a program interface, wherein the program interface, responsive to receiving a request for attributes for an entity from a client (Specification, p. 11, ll. 10-20; Figure 4, "XML Request"), generates a method call for the utility method (Specification, p. 12, ll. 3-4; Figure 4 "Call Method"). The method call identifies the entity and a response object name (Specification, p. 14, ll. 11-24; Figure 5). The program interface generates a response object (Specification, p. 14, ll. 11-24; Figure 4, "XML Response Object") and assigns the response object name to the response object (Specification, p. 14, ll. 11-24). The program interface returns the response object to the client (Specification, p. 15, ll. 2-4).

E. CLAIM 22 – INDEPENDENT

The subject matter of claim 22 is directed to a computer program product, in a computer readable physical storage medium, for code reusability and maintainability (Specification, p.3, ll.3-23; Figure 4, 422; Figure 5, 522). The computer program product includes instructions, in a utility class, for defining a utility method (Specification, p. 11, l. 10-p. 12, l. 2). The computer program product further includes instructions, responsive to receiving a request at a server for attributes for an entity from a client (Specification, p. 11, ll. 10-20; Figure 4, "XML Request"), for generating a method call for the utility method (Specification, p. 12, ll. 3-4; Figure 4 "Call Method"), wherein the method call identifies the entity and a response object name (Specification, p. 14, ll. 11-24; Figure 5). The computer program product further includes instructions for generating a response object (Specification, p. 14, ll. 11-24; Figure 4, "XML Response Object") and assigning the response object name to the response object (Specification, p. 14, ll. 11-24). The computer program product further includes instructions for returning the response object to the client (Specification, p. 15, ll. 2-4).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The ground of rejection to review on appeal is as follows:

A. GROUND OF REJECTION 1

Whether claims 1 and 3-23 fail to be anticipated under 35 U.S.C. § 102 by *Upton*, System and Method for Providing a Java Interface to an Application View Component, U.S. Patent Publication No. 2003/0110315 A1 (June 12, 2003) (hereinafter "*Upton*").

ARGUMENT

A. GROUND OF REJECTION 1 (Claims 1 and 3-23)

A.1. *Claims 1, 3-4, 6-7, and 9-23*

The Examiner rejects claims 1, 3-4, 6-7, and 9-23 under 35 U.S.C. § 102 as being anticipated by *Upton*, System and Method for Providing a Java Interface to an Application View Component, U.S. Patent Publication No. 2003/0110315 A1 (June 12, 2003) (hereinafter "*Upton*"). Appellants request that the Board of Patent Appeals and Interferences overturn this rejection.

Claim 1 is a representative claim of this grouping of claims. Claim 1 is as follows:

> 1. A method in a data processing system, for code reusability and maintainability, the method comprising:
> providing a utility class in a server, wherein the utility class defines a utility method, the utility method being written in an object oriented programming language;
> receiving a markup language request at the server for an entity from a client, the markup language request including a response object name;
> responsive to receiving the markup language request at the server for the entity from the client, generating a method call for the utility method, wherein the method call identifies the entity and the response object name;
> generating a markup language response object and assigning the response object name to the markup language response object; and
> returning the markup language response object to the client.

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). In this case, each and every feature of the presently claimed invention is not identically shown in the cited reference, arranged as they are in the claims.

With regard to claim 1, the Examiner states the following:

> Per claim 1, Upton discloses a method, in a data processing system, for code reusability and maintainability (e.g. Figure 4 and related text), the method comprising:
>
> providing a utility class in a server (e.g. Figure 3, server 304 and related text), wherein the utility class defines a utility method (para [0109] "...provide utility classes..."), the utility method being written in an object oriented programming language (para [0120-0123] "...public Class getManagedConnection Factory-Class ( )...");
>
> receiving a markup language request at the server for an entity from a client, the markup language request including a response object name (para [0076] "...request XML..." and para [0179] "...public void onAsyncServiceResponse (object asr)...";
>
> responsive to receiving the markup language request at the server for the entity from the client (para [0076] "...request XML..."), generating a method call for the utility method, wherein the method call identifies the entity and the response object name (para [0178] "...an event such as public void can be generated...");
>
> generating a markup language response object and assigning the response object name to the markup language response object (para [0174] "...a method such as public return a value..."); and
>
> returning the markup language response object to the client (para [076] "...view teh response XML..." and e.g. Figure 3, 300 and related text).

Final Office Action dated November 27, 2007, pp. 12-13.

A.1.a. *Upton* does not disclose a method, in a data processing system, for code reusability and maintainability

In support of rejecting this claim feature, the Examiner relies on *Upton's* Figure 4, and the supporting text. *Upton's* Figure 4 is a process for creating an application view. With regard to Figure 4, *Upton* states:

> Using the appropriate JSP- or Java-based interface, a Java client or Java application can create or modify an application view component using a method such as that shown by the flowchart of FIG. 4. The Java client can open the Java interface 400. The client application or user of the application can determine whether the appropriate application view exists 402. If not, the application view can be created 404, such as by using a Web application component. Once the application view exists, the user can determine whether the application view is defined 406. If not, the user can define the application view 408, such as by using a "Define App View" JSP in the Java interface. If there are services to be added 410, the user can add services 412

> such as by using an "Add Services" JSP. If there are events to be added 414, the user can add events 414 such as by using an "Add Events" JSP.

Upton, paragraph [0052].

As stated above, the application view of *Upton* provides a user interface for viewing, creating, defining, and testing the mapping of functionality from the enterprise system to the client usable format. A method for creating such a user interface is not a "a method, in a data processing system, for code reusability and maintainability," according to the preamble of claim 1.

Generally, *Upton* discloses a Java based system for viewing, creating, defining, and testing the translation from functionality of an enterprise system to a client system. *Upton* states that there are many enterprise systems that do not support web access from a client. *Upton* proposes a java based translation system that takes the antiquated enterprise system, and translates it into a format usable by the client system. The subject matter of *Upton* specifically relates to an application view. The application view of *Upton* provides a user interface that allows a person to view, create, define, and test the mapping of the functionality of the enterprise system to the client usable format.

As stated by *Upton*,

> Systems and methods in accordance with the present invention can take advantage of a Java-based interface to an application view component. An application view component can be used that provides an interface to an application or enterprise system for a Java client or Java application. A resource adapter can be used to expose functionality in the enterprise system to the Java client application. The resource adapter can be used to define services and events in the enterprise system that are available to the Java client application. A Java-based interface for the resource adapter can allow the Java client application to access the application view component. The Java-based interface can be a design-time graphical user interface, which can include a set of Java server pages and can be Web-based. The Java-based interface can allow a Java client application to access the application view component in order to accomplish a task such as creating, defining, deploying, and testing the application view component. Each of these tasks can have their own page in the interface, such as a Java server page.

Upton, paragraph [0021].

These disclosures are not the same as the claimed method for code reusability and maintainability. As further described below, these teachings are also not the same as other claimed features.

Therefore, *Upton* does not disclose all of the features of claim 1. Hence, the Examiner's rejection of claim 1 under 35 U.S.C. § 102 should be overturned.

A.1.b. *Upton* does not disclose receiving a markup language request at the server for an entity from a client, the markup language request including a response object name

The Examiner relies on *Upton's* paragraphs [0076] and [0179] as teaching this claimed feature. *Upton's* paragraph [0076] is as follows:

> A page can be provided that includes a summary of an undeployed application view. Specifically, the page can show information such as the connection criteria, a list of events, and a list of services. For each event on the application view, the user can view the appropriate XML schema, remove the event, or provide event properties. For each service on the application view, the user can view the request XML schema, view the response XML schema, or remove the service.

Upton, paragraph [0076].

Consistent with the purpose of *Upton*, paragraph [0076] states that the application view can be used to view various schema for XML documents, including a request XML schema, and a response XML schema. However, simply being able to view these schemas does not correlate to the claim 1 feature of "receiving a markup language request at the server for an entity from a client." In the cited paragraph, *Upton's* server has not received anything. Therefore, contrary to the Examiner's assertion, the cited paragraph of *Upton* does not disclose "receiving a markup language request at the server for an entity from a client" as recited in claim 1.

Applicants next address paragraph [0179] of *Upton*. *Upton's* paragraph [0179] is as follows:

> An event such as "public void onAsyncServiceResponse(Object asr) throws Exception;" can be generated whenever an application view asynchronous response is received. This method can be private, and not used by JWS clients. Here, the event can be the asynchronous response object representing the response from an asynchronous service invocation.

Upton, paragraph [0179].

The cited paragraph indicates that an "asynchronous service invocation" prompts a response of an "asynchronous response object." Neither of the terms "asynchronous service invocation" nor "asynchronous response object" is mentioned anywhere else within the reference. Despite the Examiner's assertion otherwise, neither this paragraph, nor elsewhere in *Upton*, discloses a request, wherein the request includes a name for the response object.

The Examiner has not cited any teaching wherein a markup language request is received at the server for an entity from a client. Furthermore, the Examiner has not cited any teaching wherein a received markup language request includes a response object name. In fact, *Upton* does not teach this claimed feature anywhere. Therefore, *Upton* does not disclose all of the features of claim 1. Accordingly, the Examiner's rejection of claim 1 under 35 U.S.C. § 102 should be overturned.

A.1.c. *Upton* does not disclose generating a method call [that] identifies the entity and the response object name

The Examiner relies on *Upton's* paragraph [0178] as teaching this claimed feature. *Upton's* paragraph [0178] is as follows:

> An onEvent event, such as "public void onEvent(Object event) throws Exception;" can be generated whenever an application view event is received. This method can be private, and not used by JWS clients. The "event" object can represent the event in the EIS.

Upton, paragraph [0178].

The cited paragraph indicates that receipt of an "application view event" prompts a response of an "onEvent." Neither of the terms "application view event" nor "onEvent" is mentioned anywhere else within the reference. Despite the Examiner's assertion otherwise, neither this paragraph, nor elsewhere in *Upton*, discloses generating a method call [that] identifies the entity and the response object name.

The Examiner has not cited any teaching wherein a method call [that] identifies the entity and the response object name is generated. In fact, *Upton* does not teach this claimed feature anywhere. Therefore, *Upton* does not disclose all of the features of claim 1. Accordingly, the Examiner's rejection of claim 1 under 35 U.S.C. § 102 should be overturned.

A.1.d. *Upton* does not disclose generating a markup language response object and assigning the response object name to the markup language response object

The Examiner relies on *Upton's* paragraph [0174] as teaching this claimed feature. *Upton's* paragraph [0174] is as follows:

> A method such as "public boolean isEventDeliveryEnabled( )" can return a value of "true" if event delivery is enabled, and "false" otherwise. This can be true after a call to enableEventDelivery( ) and before a call to disableEventDelivery( ).

Upton, paragraph [0174]

While the cited paragraph may show returning a response, nothing in the cited paragraph or elsewhere in *Upton* discloses that the response indicated is a markup language response object. Furthermore, nothing in the cited section, or elsewhere in *Upton* discloses that the returned response has been assigned the response object name that was previously received in the markup language request.

The Examiner has not cited any teaching wherein a markup language response object and assigning the response object name to the markup language response object is generated. In fact, *Upton* does not teach this claimed feature anywhere. Therefore, *Upton* does not disclose all of the features of claim 1. Accordingly, the Examiner's rejection of claim 1 under 35 U.S.C. § 102 should be overturned.

A.2. *Claim 5*

Claim 5 is representative of the group. Claim 5 is as follows:

> 5. The method of claim 1, further comprising:
> retrieving, by the utility method, at least one data item for the method call and the entity,
> wherein the response object includes the at least one data item.

With regard to claim 5, the Examiner states the following:

> Per claim 5, Upton discloses the method of claim 1, further comprising: retrieving, by the utility method, at least one data item for the method call and the entity, wherein the response object includes the at least one data item (para [0082] "...message bundle to retrieve... message...").

Final Office Action dated November 27, 2007, p. 3.

The Examiner relies on *Upton's* paragraph [0082] as teaching this claimed feature. *Upton's* paragraph [0082] is as follows:

> An ActionResult class can encapsulate information about the outcome of processing a request. ActionResult can also provide information to ControllerServlet to help determine the next page to display to the user. A Word class, as well as its descendants, can supply logic to validate form fields, as all fields in a web application can require some validation. If any fields are invalid, a Word object can use a message bundle to retrieve an internationalized/localized error message for the field.

Upton, paragraph [0082].

Consistent with the other teaching of *Upton*, paragraph [0082] discloses a form processing class for use in the java based user interface application view. As disclosed therein, if any form fields are determined invalid, a Word object can use a message bundle to retrieve an internationalized/localized error message for the field.

In rejecting claim 1, the Examiner rejected the Appellant's "response object" feature of claim 1 based on *Upton's* paragraph [0174]. Assuming *ad arguendum*, that the Examiner is correct, despite the Appellants showing of the contrary, the Examiner's equivalent of the Appellant's "at least one data item" must be found within the Examiner's equivalent of the Appellant's "response object." The Examiner makes no attempt to explain any relation between the two cited sections, and indeed *Upton* does not disclose one. Paragraphs [0174] and [0082] disclose completely different functionalities.

Even if paragraph [0082] discloses "at least one data item," there is no indication in *Upton* that any "at least one data item" disclosed in paragraph [082] are included in the "response object." of paragraph [0174]. Therefore, even if paragraph [0082] discloses an "at least one data item," such "at least one data item" does not anticipate the recited feature of claim 8.

The Examiner has not cited any teaching wherein the response object includes the at least one data item. In fact, *Upton* does not teach this claimed feature anywhere. Therefore, *Upton* does not disclose all of the features of claim 5. Accordingly, the Examiner's rejection of claim 8 under 35 U.S.C. § 102 should be overturned.

A.3. *Claim 8*

Claim 8 is representative of the group. Claim 8 is as follows:

> 8. The method of claim 5, wherein the request includes a list of attributes.

With regard to claim 8, the Examiner states the following:

> Per claim 8, Upton discloses the method of claim 5, wherein the request includes a list of attributes (para [0127] "...required attribute...").

Final Office Action dated November 27, 2007, p. 4.

The Examiner relies on *Upton's* paragraph [0127] as teaching this claimed feature. *Upton's* paragraph [0127] is as follows:

> A user can display the label for the form field using a line such as: <adk:label name=`eventName` required=`true`/>. This line can display a label for a field on the form. The value that is displayed can be retrieved from the message bundle for the user. The "required" attribute can indicate whether the user must supply this parameter to be successful.

Upton's paragraph [0127]

In rejecting claim 1, the Examiner equated the Appellant's request to the "request XML schema" of *Upton's* paragraph [0076]. Assuming *ad arguendum*, that the Examiner is correct, despite the Appellants showing of the contrary, the "request XML schema" of *Upton* must include a list of attributes in order to maintain the Examiner's rejection. The Examiner provides no citation or teaching that would indicate as such.

Even if paragraph [0127] discloses a "list of attributes," there is no indication in *Upton* that any attributes disclosed in paragraph [0127] are included in the "request XML schema" paragraph [0076]. Therefore, even if paragraph [0127] discloses a "list of attributes," such a list does not anticipate the recited feature of claim 8.

The Examiner has not cited any teaching wherein the request includes a list of attributes. In fact, *Upton* does not teach this claimed feature anywhere. Therefore, *Upton* does not disclose all of the features of claim 8. Accordingly, the Examiner's rejection of claim 8 under 35 U.S.C. § 102 should be overturned.

C. CONCLUSION

As shown above, the examiner has failed to state valid rejections against any of the claims. Therefore, Applicants request that the Board of Patent Appeals and Interferences reverse the rejections. Additionally, Applicants request that the Board direct the examiner to allow the claims.

/Brandon G. Williams/
Brandon G. Williams
Reg. No. 48,844
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 385-8777

CLAIMS APPENDIX

The text of the claims involved in the appeal is as follows:

1. A method, in a data processing system, for code reusability and maintainability, the method comprising:

providing a utility class in a server, wherein the utility class defines a utility method;

responsive to receiving a request at the server for attributes for an entity from a client, generating a method call for the utility method, wherein the method call identifies the entity and a response object name;

generating a response object and assigning the response object name to the response object; and

returning the response object to the client.

3. The method of claim 1, wherein the request is an extensible markup language request.

4. The method of claim 3, wherein the extensible markup language request is one of a list request and a get request.

5. The method of claim 1, further comprising:

retrieving, by the utility method, at least one data item for the method call and the entity, wherein the response object includes the at least one data item.

6. The method of claim 5, wherein the step of retrieving at least one data item includes retrieving the at least one data item from a database.

7. The method of claim 6, wherein the at least one data item is retrieved from the database through a structured query language interface.

8. The method of claim 5, wherein the request includes a list of attributes.

9. The method of claim 8, wherein the at least one data item includes a set of attributes for the entity, wherein the set of attributes corresponds to the list of attributes.

10. The method of claim 9, wherein the list of attributes is an empty string.

11. The method of claim 10, wherein the set of attributes includes all attributes for the entity.

12. The method of claim 1, wherein the response object is an extensible markup language document.

13. An apparatus, in a data processing system, for code reusability and maintainability, the apparatus comprising:

a utility class, wherein the utility class defines a utility method;

a program interface, wherein the program interface, responsive to receiving a request for attributes for an entity from a client, generates a method call for the utility method, wherein the method call identifies the entity and a response object name;

wherein the program interface generates a response object and assigns the response object name to the response object; and

wherein the program interface returns the response object to the client.

14. The apparatus of claim 13, wherein the client includes an extensible markup language interface and wherein the request is an extensible markup language request.

15. The apparatus of claim 13, wherein the utility method retrieves at least one data item for the method call and the entity and wherein the response object include the at least one data item.

16. The apparatus of claim 15, wherein the utility method retrieves the at least one data item from a database.

17. The apparatus of claim 15, wherein the request includes a list of attributes.

18. The apparatus of claim 17, wherein the at least one data item includes a set of attributes for the entity, wherein the set of attributes corresponds to the list of attributes.

19. The apparatus of claim 18, wherein the list of attributes is an empty string.

20. The apparatus of claim 19, wherein the set of attributes includes all attributes for the entity.

21. The apparatus of claim 13, wherein the response object is an extensible markup language document.

22. A computer program product, in a computer readable physical storage medium, for code reusability and maintainability, the computer program product comprising:

instructions, in a utility class, for defining a utility method;

instructions, responsive to receiving a request at a server for attributes for an entity from a client, for generating a method call for the utility method, wherein the method call identifies the entity and a response object name;

instructions for generating a response object and assigning the response object name to the response object; and

instructions for returning the response object to the client.

23. The method of claim 1, wherein the server is located at a first computer system, wherein the client is located at a second computer system, and wherein the first computer system is separate from the second computer system.

<u>EVIDENCE APPENDIX</u>

This appeal brief presents no additional evidence.

RELATED PROCEEDINGS APPENDIX

This appeal has no related proceedings.